A novel approach for analysis of attack graph

Yousefi, Mohammadmehdi; Mtetwa, Nhamoinesu; Zhang, Yan; Tianfield, Huaglory

Published in:
2017 IEEE International Conference on Intelligence and Security Informatics: Security and Big Data, ISI 2017

DOI:
10.1109/ISI.2017.8004866

Publication date:
2017

Document Version
Author accepted manuscript

A Novel Approach for Analysis of Attack Graph

Mehdi Yousefi, Nhamo Mtetwa, Yan Zhang, and Huaglory Tianfield
CCIS Department, Glasgow Caledonian University
Glasgow, United Kingdom
E-mail: {mehdi.yousefi, nhamoinesu.mtetwa, yan.zhang, h.tianfield}@gcu.ac.uk

Abstract— Attack graph technique is a common tool for the evaluation of network security. However, attack graphs are generally too large and complex to be understood and interpreted by security administrators. This paper proposes an analysis framework for security attack graphs for a given IT infrastructure system. First, in order to facilitate the discovery of interconnectivities among vulnerabilities in a network, multi-host multi-stage vulnerability analysis (MulVAL) is employed to generate an attack graph for a given network topology. Then a novel algorithm is applied to refine the attack graph and generate a simplified graph called a transition graph. Next, a Markov model is used to project the future security posture of the system. Finally, the framework is evaluated by applying it on a typical IT network scenario with specific services, network configurations, and vulnerabilities.

Keywords— cyber security; security metrics; vulnerability assessment; attack graph

I. INTRODUCTION

We live in a connected digital world whose connectivity is provided by enterprise networks that grow both in size and complexity [1]. The security of systems and services that rely on these networks for day-to-day operations is critical. Vulnerabilities exist in these enterprise networks due to weaknesses in technical design, configuration, and security policies. These vulnerabilities expose the networks to severe risks. Attackers can take advantage of these vulnerabilities and compromise the systems for malicious purposes.

Securing enterprise networks against security threats is an active area of research. Traditional methods based on instinct and experience is not sufficient for network protection because security management is becoming more and more complex and sophisticated. Almost everyday new vulnerabilities emerge making networks more vulnerable to intruders. The combination of different vulnerabilities enables attackers to stage multi-step attacks that are very difficult to identify and defend against [2]. These attacks can be highly sophisticated and current security countermeasure techniques may not be able to handle their complexity.

In this paper, a metric based security framework is proposed to evaluate the overall security impact of software vulnerabilities on a given network. First, we employ MulVAL [3] to find all the attack paths in the network. Second, we propose an algorithm to refine the attack graph to produce a transition graph which is used in the analysis part of the framework and makes it easy to understand and analyze the attack graph for monitoring purposes. We then use CVSS[1] to calculate transition probabilities for the transition graph [1] and apply Absorbing Markov Chain to predict attacker behavior [4].

The rest of the paper is organized as follows: in section II, we review some work in this field. In section III, the details of the framework and network topology and algorithm are presented. In section IV, we give an example to evaluate the framework. Finally, in section V, we compare the work with similar work in the field and conclude the paper.

II. LITERATURE REVIEW

Cyber vulnerability assessment is "the process of identifying the vulnerabilities in a system and prioritizing them according to their severity"[2]. It helps to identify these weaknesses in order to apply proper patches [5]. In [6-8], the authors provide a standard security evaluation baseline which does not include quantitative measurements. To address this problem in [9], CVSS is developed by NIST as a standard composite scoring system model which is usable and understandable by security practitioners. It considers each of vulnerabilities as an isolated entity. If we have multiple vulnerabilities, CVSS has no foresight of exploitable possible interrelationships between the vulnerabilities.

Further work was done in [10-19], to address the aforementioned problem. The attack graph technique is a common tool for the evaluation of network security. It has been developed to automatically identify multi-stage attacks in enterprise networks.

In [3, 15, 19-24], the authors focus on attack graph generation and improving the complexity of employed algorithms. The main advantages of these research works are that they consider interrelationships between vulnerabilities in the enterprise network. However, attack graphs are too large and complex to be understood and interpreted by security administrators. To address this problem various approaches have been introduced in [17, 26-29]. These approaches try to improve the visualization of an attack graph through abstraction [17], data reduction [25], and user interaction [29].

Different attack graph methods draw all the possible attack paths, but still, it is necessary to combine them with a sort of metric to measure the validity and possibility of each path. In other words, there is a need to establish security frameworks that are able to measure the security risk in enterprises objectively.

[1] https://www.first.org/cvss

[2] https://oval.mitre.org/

III. Security Framework using Attack Graph and Markov Model

The proposed framework includes the following steps. The first step is to collect information about the current state of the network. This includes a list of known vulnerabilities and the configuration of the network including services and access rules. "Open Vulnerability Assessment Language" (OVAL) is used to gather information about vulnerabilities and services in the network. The second step is to identify the correlations between vulnerabilities by generating attack graph. It helps to consider the interdependencies between vulnerabilities in the system based on network connectivity. MulVAL is used for generating the attack graph. MulVAL is open source and the complexity of the attack graph generation grows between $O(n^2)$ and $O(n^3)$ [2]. Attack graphs are important tools to analyze the security of the network based on existing vulnerabilities [1-3]. Then we refine the attack graph and generate transition graph. The final step is to use the transition graph to apply the Absorbing Markov Chain which ranks the vulnerabilities in terms of importance.

A. Algorithm to Refine Attack Graph

Attack graph can be too large and complex. We propose an algorithm to refine the attack graph into a transition graph. This graph shows all the possible attacker's movements between vulnerabilities and it is easy to understand and interpret. Details of our algorithm are described in Algorithm and Processes 1- 4. The interrelationship between processes is shown in Fig.1.

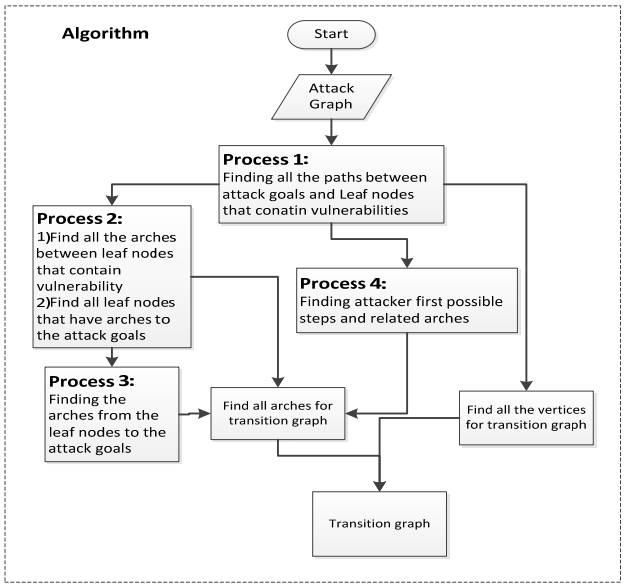

Fig. 1. Flowchart of the proposed Algorithm

The output of MulVAL includes 'input.txt','Vertices.CSV', 'ARCS.CSV', and 'AttackGraph.txt'. The 'input.txt' is the input file for the MulVAL and includes the configuration of the network including the network services, network connectivity and a list of vulnerabilities. The 'Vertices.CSV' includes all vertices in attack graph and the 'ARCS.CSV' includes all the edges between vertices. The 'AttackGraph.txt' is attack graph in text format.

In Algorithm, Vt (line 16) is the vertices of the transition graph and Et (line 15) is edges for the transition graph.

Algorithm Refining the attach graph

```
1:  procedure transitionGraph
2:      lineofinputFile = Read(input.txt)
3:      linesofArcsFile = Read(ARCS.CSV)
4:      linesofVerticesFile = Read(Vertices.CSV)
5:      linesofAttackFile = Read(AttackGraph.txt)
6:      V = defineVertices(linesofVerticesFile)
7:      E = defineEdges(linesofArcsFile)
8:      attackGoals = defineRootNodes(linesofAttackFile)
9:      leaves = defineLeaves(linesofVerticesFile)
10:     attackerLocation = defineAttackerLoc(linesofVerticesFile)
11:     G = defineDirectedGraph(V, E)
12:     pathsVt = definePathsvulExists(G,attackGoals, leaves)
13:     pIncludeVstoGoals = defineArchsvPaths(pathsVt) #tempPath
14:     edgestoNGoalV = defineArchsvPaths(pathsVt) #et
15:     Et = edgestoNGoalV +
        defineArchstoGoals(pIncludeVstoGoals,
        attackGoals) + defineAttackerFirst Steps(linesOfVerticesFile ,
        pathsVt, attackerLocation)
16:     Vt = attackerLocation + attackGoals + {leaves if vulExist =
        True}
17:     G2 = nx.DiGraph()
18:     G2.add_nodes_from(Vt)
19:     G2.add_edges_from(Et)
20:     showGraph(G2)
21: End procedure
```

Process 1 (line 12, Algorithm, definePathsvulExists()) returns all the paths between attack goals and leaves which include vulnerabilities (vulExist = True). In this process, we use an improved depth first search algorithm from networkx library. It generates all simple paths in the graph from source to target vertices (Process 1, line 5, all_simple_paths(G, source, target)).

Process 1 Finding the paths between attack goals and leaves that include vulnerability

```
1:  procedure definePathsvulExists(G, attackGoals, leaves)
2:      allPaths = Ø
3:      For i = 1 to length(attackGoals) do
4:          For j= 1 to length(leaves) do
5:              allPaths = allPaths + nx.all_simple_paths(G,
                leaves[j], attackGoals [i]))
6:          End For
7:      End For
8:      vPaths = { ∀ Path ∈ allPaths | path.vulExists =
        True}
9:  End procedure
```

Process 2 (Algorithm, line 13, 14) returns edges between leaf nodes that contain a vulnerability and it also returns paths that include nodes with an edge to the attack goals.

Process 2 Finding edges between leaves with vulExists = True

```
1:  procedure defineArchsvPaths(pathsVt)
2:      et = Ø
3:      tempPaths = pathsVt
4:      For i=1 to length(vPaths) do
5:          X = vPaths.remove(vPaths(i))
6:          For j=1 to length(X) do
```

```
7:                If vPaths[i] ⊆ X[j] then
8:                    et = et + (X[j].leaf.number ,vPaths[i].leaf.number)
9:                    tempPath = tempPath.remove(X[j])
10:               End If
11:           End For
12:       End For
13:       return et, tempPath
14:   End procedure
```

Process 3 (Algorithm, line 15), uses the second output of Process 2 and finds edges to the attack goals.

Process 3 Finding edges to the attack goals

```
1:  procedure defineArchstoGoals(pIncludeVstoGoals, attackGoals)
2:      et = Ø
3:      For i=1 to length(pIncludeVstoGoals) do
4:          For j=1 to length(attackGoals) do
5:              If j != length(attackGoals)Then
6:                  If tempPaths[i].leaf.number > attackGoals[j] and
                        tempPaths[i].leaf.number < attackGoals[j+1]
                        then
7:                      et = et + (tempPaths[i].leaf, attackGoals[j])
8:                  End If
9:              Else If j == length(attackGoals) then
10:                 et = et + (tempPaths[i].leaf, attackGoals[j])
11:             End If
12:             End If
13:         End For
14:     End For
15: End procedure
```

Process 4 (Algorithm, line 15), finds the attacker's first possible steps and related edges for the transition graph.

Process 4 Finding the attacker first possible steps and related edges for the transition graph

```
1:  procedure defineAttackerFirstSteps(linesOfVerticesFile, pathsVt,
    attackerLocation)
2:      For all item in linesOfVerticesFile do
3:          If item includes "hacl" and "internet" then
4:              tempList = tempList + item
5:          End If
6:      End For
7:      For all item in tempList do
8:          O1 =find the first occurrence of '('
9:          O2 = find the third occurrence of ','
10:         tempList2 = tempList2 + string between O1 and O2
11:     End For
12:     For all item in tempList2 do
13:         For item2 in pathsVt do
14:             If item2.find(item)Then
15:                 firstSteps = firstSteps+item.number
16:             End If
17:         End For
18:     End For
19:     For all item in firstSteps do
20:         firstPSteps= firstPSteps + (attackerLocation , item)
21:     End For
22: End procedure
```

B. Applying Markov Model to Predict Attacker Behaviour

Reinforcement learning technique can be used for quantitative security evaluation of large-scale enterprise networks. One such reinforcement learning technique is Absorbing Markov Chains. This model can be applied to problems that satisfy two conditions: 1) the problem must have at least one absorbing state. A state is called absorbing if once that state is entered, it is impossible to leave. 2) From each non-absorbing state (transient state), it must be possible to go to an absorbing state with finite steps. In attack graph, vulnerabilities are considered as states. If a vulnerability is not an ultimate attck goal, the attacker use it as stepping stone to reach the ultimate goal. This vulnerability is considered as a non-absorbing state. Absorbing states are the ultimate attacker's goals. When attacker compromises the goal state, they will stay there till the administrator applies proper countermeasures. In this work we apply Absorbing Markov Chain [4] assuming the attacker is located on the Internet and chooses vulnerabilities to reach their goal based on maximizing the probability of their success.

A transition matrix for an absorbing Markov Chain is a standard form if the rows and columns are labeled so that all the non-absorbing states precede all the absorbing states. Any standard form can be partitioned into four sub-matrices:

$$P = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix} \quad (1)$$

Here Q is a matrix of transition states; R is a matrix of absorbing states, I is the identity matrix and 0 is a zero matrix. For an absorbing Markov chain, the matrix N or so-called Fundamental matrix provides a valuable insight to predict future attacker's behavior. The matrix N is the inverse of matrix $I - Q$:

$$N = (I - Q)^{-1} \quad (2)$$

Here n_{ij} of N gives the expected number of times an attacker visited transient state s_j if the attacker starts in the transient state s_i.

Let t_i be the expected number of steps that attacker needs to take before compromising the security of the network; given the fact that attacker starts in state s_j.

$$t = N \times c \quad (3)$$

Where c is a column vector all of whose entries are 1. Let b_{ij} be the probability that an attacker will compromise the network in state s_j if attacker starts in the transient state s_i. Let B be the matrix with entries b_{ij} then

$$B = N \times R \quad (4)$$

IV. EXAMPLE TO EVALUATE THE FRAMEWORK

A. Topology and System Configuration

The network topology is shown in Fig. 2. There are five different services including a web server and a file server in the same subnet, a Citrix and a VPN server in the same subnet and a Comm server and a dataHistorian excel in the same subnet. It has also two workstations running Acrobat and Internet Explorer. There are two firewalls including perimeter and internal firewalls. The network connectivity in this topology is based on firewall rules described below. (i)The attacker is located on the Internet and has access to the servers as follows: the web server, VPN server, Citrix server and Comm server through the HTTP protocol and HTTP port. (ii)There is

bidirectional connectivity between web server and other machines including file server, work station through the HTTP protocol and HTTP port. (iii)There is bidirectional connectivity between VPN server and other machines including Citrix server, work station through the HTTP protocol and HTTP port. (iv)There is a bidirectional access between Comm server and data Historian through HTTP protocol and HTTP port. (v)The file Server and workstation have access to each other through NFS protocol and NFS port.

There are four different vulnerabilities in the topology Fig. 2. Vulnerabilities are recognized by unique identifiers. These identifiers are assigned by the National Vulnerability Database (NVD)[3].

The Citrix server contains a vulnerability named 'CVE-2010-0490'. This is a weakness in Internet Explorer (IE) 6, 7 and 8 with the possibility that remote intruder can execute arbitrary code on the target machine. The VPN Server contains 'CVE-2010-0492' that is related to IE 8 with the possibility that attacker can execute arbitrary code on the target machine. The Comm Server contains 'CVE-2010-0483' which is related to VBScript in Windows. When IE is used the attacker can execute arbitrary code on the target machine. The dataHistorian mainframe contains 'CVE-2010-0494'. This vulnerability is

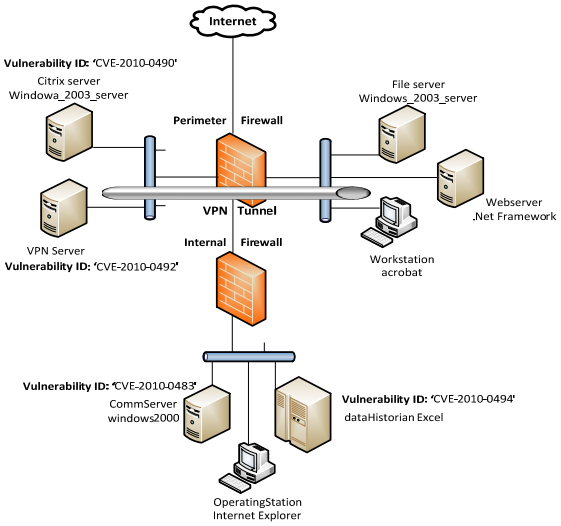

Fig. 2. Network Topology

related to IE 6, 7, and 8. The attacker can conduct Cross-site scripting attack. Table 1 summarized the vulnerabilities in the system.

Table 1: Vulnerabilities in the Topology

List of Vulnerabilities	Vulnerabilities and Associated Machines	
	Machine	***Vulnerability ID (NVD)***
1	citrixServer	CVE-2010-0490
2	vpnServer	CVE-2010-0492
3	commServer	CVE-2010-0483
4	dataHistorian Excel	CVE-2010-0494

[3] https://nvd.nist.gov/

B. Generating Attack Graph and Transition Graph

We use MulVAL to generate the attack graph. Fig. 3 shows the attack graph along with description of nodes in the graph. There are three different vertices in the attack graph. The square vertices (e.g. nodes 5, 16 and 32 in Fig. 3) are related to system configuration. For example firewall rules that let web server be accessible from the Internet or the buggy software on a machine. The diamond vertices (e.g. nodes 3, 12, and 25 in Fig. 3) represent potential privileges or access that an attacker can obtain in the system, e.g., code execution privilege on the web server. The elliptical vertices (e.g. nodes 4, 11, and 26 in Fig. 3) link preconditions to postconditions. As an example, it is necessary for an attacker to have access to a machine that has a vulnerability, to be able to exploit the vulnerability and obtain privileges.

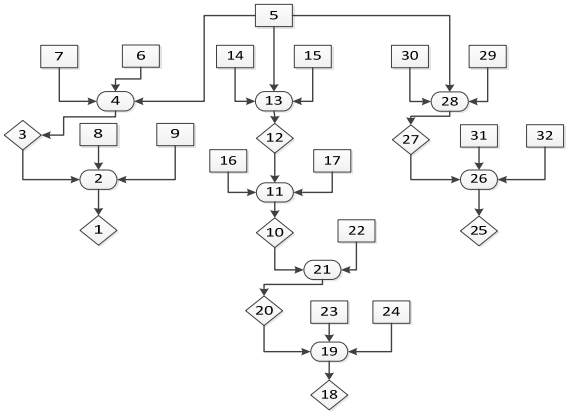

Fig. 3. Attack Graph

```
1,"execCode(citrixServer,user)","OR",0
2,"RULE 3 (remote exploit for a client program)","AND",0
3,"accessMaliciousInput(citrixServer,victim_2,ie)","OR",0
4,"RULE 22 (Browsing a malicious website)","AND",0
5,"attackerLocated(internet)","LEAF",1
6,"hacl(citrixServer,internet,httpProtocol,httpPort)","LEAF",1
7,"inCompetent(victim_2)","LEAF",1
8,"hasAccount(victim_2,citrixServer,user)","LEAF",1
9,"vulExists(citrixServer,'CVE-2010-
    0490',ie,remoteClient,privEscalation)","LEAF",1
10,"execCode(commServer,user)","OR",0
11,"RULE 3 (remote exploit for a client program)","AND",0
12,"accessMaliciousInput(commServer,victim_1,windows_2000)","OR",0
13,"RULE 22 (Browsing a malicious website)","AND",0
14,"hacl(commServer,internet,httpProtocol,httpPort)","LEAF",1
15,"inCompetent(victim_1)","LEAF",1
16,"hasAccount(victim_1,commServer,user)","LEAF",1
17,"vulExists(commServer,'CVE-2010-
    0483',windows_2000,remoteClient,privEscalation)","LEAF",1
18,"execCode(dataHistorian,root)","OR",0
19,"RULE 2 (remote exploit of a server program)","AND",0
20,"netAccess(dataHistorian,httpProtocol,httpPort)","OR",0
21,"RULE 5 (multi-hop access)","AND",0
22,"hacl(commServer,dataHistorian,httpProtocol,httpPort)","LEAF",1
23,"networkServiceInfo(dataHistorian,mountd,
    httpProtocol,httpPort,root)","LEAF",1
24,"vulExists(dataHistorian,'CVE-2010-
    0494',mountd,remoteExploit,privEscalation)","LEAF",1
25,"execCode(vpnServer,user)","OR",0
26,"RULE 3 (remote exploit for a client program)","AND",0
27,"accessMaliciousInput(vpnServer,victim_5,openvpn)","OR",0
28,"RULE 22 (Browsing a malicious website)","AND",0
29,"hacl(vpnServer,internet,httpProtocol,httpPort)","LEAF",1
30,"inCompetent(victim_5)","LEAF",1
31,"hasAccount(victim_5,vpnServer,user)","LEAF",1
32,"vulExists(vpnServer,'CVE-2010-0492',openvpn,remoteClient,
    privEscalation)","LEAF",1
```

This attack graph is refined with the proposed algorithm and the result is presented in Fig. 4. As it can be seen, the transition graph only includes the attacker location, vertices that contain vulnerabilities and attack goals. This simplified graph can be easily interperted by security administrators. We use this graph to generate the transition matrix for absorbing markov chain.

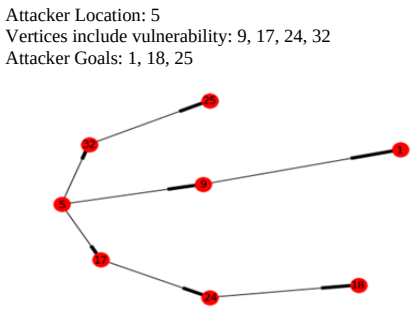

Fig. 4. Refined Attack Graph

C. *Results and Analysis*

To analyze the security of the topology in Fig. 2, the following questions are of interest. If the attacker starts in a specific transient state, then:
First, what is the number of times that the attacker visits each transient state? Second, what is the number of steps before the attacker compromise the network? Third, what is the probability that the attacker reach their goal if they start from a specific state?

Fig. 5 shows the number of times the attacker visited a transient state when they start from a specific transient state. As it can be seen if an attacker is in S5 (Internet), attacker visit S9, 0.28 times, S17, 0.16 times, and 0.44 and 0.28 times S24 and S32 respectively. Here, based on the result, S24 is visited more frequently by the attacker and the security administrator should address this vulnerability with higher priority. If the attacker is in S9, S24, and S32, it is not possible to go to any other transient state except one of the goal states. For S17 the attacker can visit S24 once. It is because, if the attacker aims to reach the attack goal, first they should compromise S24.

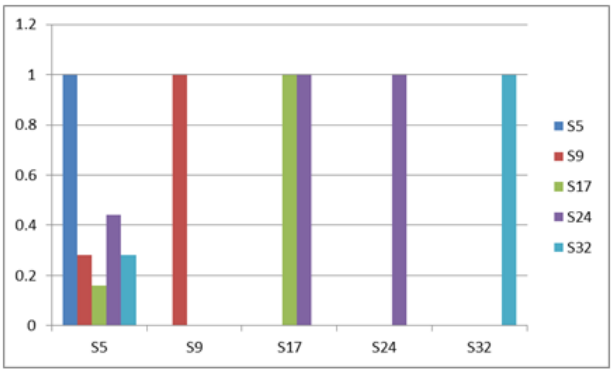

Fig. 5. Number of times attacker visits each transient state

Fig. 6 demonstrates the number of steps the attacker takes from a transient state to compromise one of the attack goals. If the attacker starts from S5, the numbers of steps are 2.16 to reach one of the attack goals. If the attacker goes to S9, or S24 or S32 from S5, then the number of steps is 1 to compromise the attack goals. But, if the attackers go to S17 from S5, then it takes 2 steps to reach the attack goal. Based on the results, the way to S17 takes longer to compromise the network. Therefore, vulnerabilities in S9, S24, and S32 have higher priority to tackle.

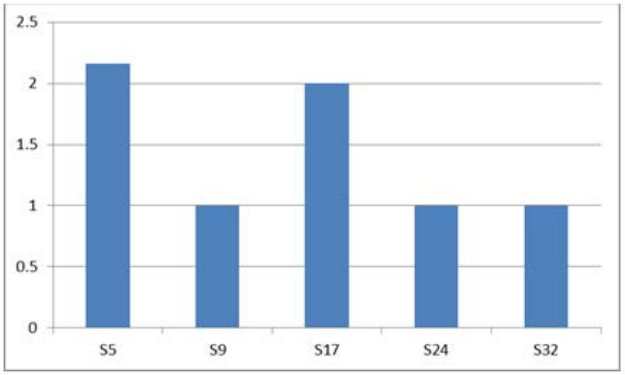

Fig. 6. Steps are taken until the attacker compromises the network

Fig. 7 shows the probability of absorption in one of the attack goals. If the attacker starts from S5, there are three different options to compromise the network including S1, S18, and S25. Attacker reaches S1 or S25 with the probability of 0.28 (28%) and to S18 with the probability of 0.44 (44%). If the attacker chooses to go from S5 to S9, the only reachable attack goal from S9 is S1; therefore, the probability of absorption in S1 is 1(100%). It is the same for other states as eventually all the states will be absorbed in one of the attack goals.

Based on what we have observed, the result provides valuable information for security administrators to make informed decisions. For example, in our experiment it takes more steps to compromise S18, but, attackers spent more time in S24 which is the only way to reach S18. Also, the probability of absorption when attacker wants to compromise S18 is highest among other attack goals. Based on what we have discussed even reaching S18 needs more steps but it is easier for the attacker to compromise S18.

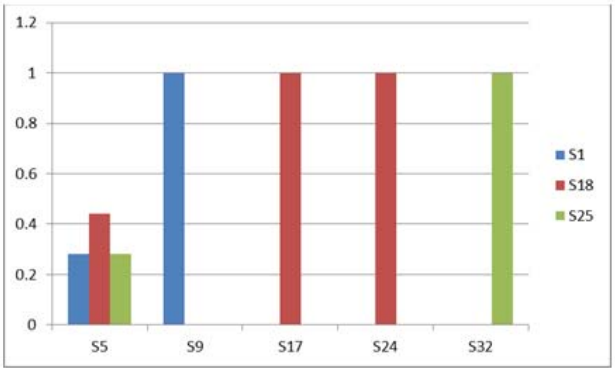

Fig. 7. Probability of absorption in one of the attack Goals

V. Discussion And Conclusion

There are approaches to reduce the complexity of attack graph [17, 25-29]. These approaches try to improve the visualization of attack graph through abstraction, data reduction, hierarchical aggregation and user interaction. Compared to these works our algorithm simply removes all the redundant information from the attack graph and only keeps minimum necessary information which is the nodes containing vulnerabilities and feasible interconnectivity between them. This simplification makes our approach suitable for monitoring tools and it can be augmented by parallel processing to scale to large enterprise networks. Another advantage of this simplified graph is that it could be used as transition graph for analytics techniques like Markov Model. In [4], the Absorbing Markov Chain is used for analytics and their focus is on employing a temporal metric to make their framework a dynamic security analysis, but, in this paper, our focus mostly is on the algorithm to refine the attack graph.

In this work, we have proposed a framework to assess the security of the computer networks. The main contribution of this paper is a novel algorithm to refine the attack graph. Attack graphs inherently are large and complex and difficult to interpret. This algorithm simplifies attack graph and makes it suitable for monitoring purposes in security operation centers and it could be used to generate transition matrix for Markov Model.

References

[1] X. Ou and A. Singhal, Quantitative security risk assessment of enterprise networks, 1st ed. New York: Springer, 2012.

[2] Ou, X., Boyer, W. F., & McQueen, M. A. (2006, October). A scalable approach to attack graph generation. In Proceedings of the 13th ACM conference on Computer and communications security (pp. 336-345). ACM.

[3] Ou, X., Govindavajhala, S., & Appel, A. W. (2005, July). MulVAL: A Logic-based Network Security Analyzer. In USENIX security.

[4] Abraham, S., & Nair, S. (2015). A predictive framework for cyber security analytics using attack graphs. arXiv preprint arXiv:1502.01240.

[5] Kissel, R. (Ed.). (2011). Glossary of key information security terms. Diane Publishing.

[6] Ferraiolo, K. (2000). The Systems Security Engineering Capability Maturity Model.

[7] Stoneburner, G., Hayden, C., & Feringa, A. (2001). Engineering principles for information technology security (a baseline for achieving security). BOOZ-ALLEN AND HAMILTON INC MCLEAN VA.

[8] Grance, T., Hash, J., Stevens, M., O'Neal, K., & Bartol, N. (2003). SP 800-35. Guide to Information Technology Security Services.

[9] https://www.first.org/cvss. Web page accessed on May 15, 2015

[10] Dawkins, J., & Hale, J. (2004, April). A systematic approach to multi-stage network attack analysis. In Information Assurance Workshop, 2004. Proceedings. Second IEEE International (pp. 48-56). IEEE.

[11] Dewri, R., Poolsappasit, N., Ray, I., & Whitley, D. (2007, October). Optimal security hardening using multi-objective optimization on attack tree models of networks. In Proceedings of the 14th ACM conference on Computer and communications security (pp. 204-213). ACM.

[12] Homer, J., Zhang, S., Ou, X., Schmidt, D., Du, Y., Rajagopalan, S. R., & Singhal, A. (2013). Aggregating vulnerability metrics in enterprise networks using attack graphs. Journal of Computer Security, 21(4), 561-597.

[13] Ingols, K., Lippmann, R., & Piwowarski, K. (2006, December). Practical attack graph generation for network defense. In Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual (pp. 121-130). IEEE.

[14] Jajodia, S., & Noel, S. (2010). Advanced cyber attack modeling analysis and visualization. GEORGE MASON UNIV FAIRFAX VA.

[15] Jajodia, S., Noel, S., & O'Berry, B. (2005). Topological analysis of network attack vulnerability. In Managing Cyber Threats (pp. 247-266). Springer US.

[16] Li, W., Vaughn, R. B., & Dandass, Y. S. (2006). An approach to model network exploitations using exploitation graphs. Simulation, 82(8), 523-541.

[17] Lippmann, R. P., & Ingols, K. W. (2005). An annotated review of past papers on attack graphs (No. PR-IA-1). MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB.

[18] Saha, D. (2008, October). Extending logical attack graphs for efficient vulnerability analysis. In Proceedings of the 15th ACM conference on Computer and communications security (pp. 63-74). ACM.

[19] Sheyner, O., Haines, J., Jha, S., Lippmann, R., & Wing, J. M. (2002). Automated generation and analysis of attack graphs. In Security and privacy, 2002. Proceedings. 2002 IEEE Symposium on (pp. 273-284). IEEE.

[20] Jajodia, S., & Noel, S. (2010). Topological vulnerability analysis. In Cyber situational awareness (pp. 139-154). Springer US.

[21] Ritchey, R. W., & Ammann, P. (2000). Using model checking to analyze network vulnerabilities. In Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on (pp. 156-165). IEEE.

[22] Ammann, P., Wijesekera, D., & Kaushik, S. (2002, November). Scalable, graph-based network vulnerability analysis. In Proceedings of the 9th ACM Conference on Computer and Communications Security (pp. 217-224). ACM.

[23] Lippmann, R., Ingols, K., Scott, C., Piwowarski, K., Kratkiewicz, K., Artz, M., & Cunningham, R. (2006, October). Validating and restoring defense in depth using attack graphs. In Military Communications Conference, 2006. MILCOM 2006. IEEE (pp. 1-10). IEEE.

[24] Noel, S., & Jajodia, S. (2005, December). Understanding complex network attack graphs through clustered adjacency matrices. In Computer Security Applications Conference, 21st Annual (pp. 10-pp). IEEE.

[25] Homer, J., Varikuti, A., Ou, X., & McQueen, M. (2008). Improving attack graph visualization through data reduction and attack grouping. Visualization for computer security, 68-79.

[26] Huang, H., Zhang, S., Ou, X., Prakash, A., & Sakallah, K. (2011, December). Distilling critical attack graph surface iteratively through minimum-cost sat solving. In Proceedings of the 27th Annual Computer Security Applications Conference (pp. 31-40). ACM.

[27] Lippmann, R. P., Ingols, K. W., Scott, C., Piwowarski, K., Kratkiewicz, K., Artz, M., & Cunningham, R. (2005). Evaluating and strengthening enterprise network security using attack graphs. Lexington, Massachusetts October.

[28] Noel, Steven, and Sushil Jajodia. "Managing attack graph complexity through visual hierarchical aggregation." Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security. ACM, 2004.

[29] Williams, L., Lippmann, R., & Ingols, K. (2008). An interactive attack graph cascade and reachability display. In VizSEC 2007 (pp. 221-236). Springer Berlin Heidelberg.